



Sortowanie przez scalanie w języku C++

- [Wprowadzenie](#)
- [Film samouczek](#)
- [Prezentacja multimedialna](#)
- [Sprawdź się](#)
- [Dla nauczyciela](#)



Sortowanie przez scalanie w języku C++

Źródło: Xavi Cabrera, domena publiczna.

Poznaliśmy już rekurencyjny algorytm [sortowania przez scalanie](#), wykorzystujący metodę „dziel i zwyciężaj”. Jest on jednym z najszybszych sposobów sortowania.

W tym e-materiale zajmiemy się implementacją algorytmu *merge sort* w języku C++.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- [Sortowanie przez scalanie w języku Java](#),
- [Sortowanie przez scalanie w języku Python](#).

Więcej zadań? Sięgnij do: [Sortowanie przez scalanie – zadania maturalne](#).

Twoje cele

- Przeanalizujesz, czym jest rekurencja oraz metoda „dziel i zwyciężaj”.
- Scharakteryzujesz działanie algorytmu sortowania przez scalanie.
- Napiszesz program wykorzystujący ten algorytm.

Film samouczek

Problem 1

Napisz program niemalejąco sortujący wyniki ankiety dotyczącej równości w życiu społecznym, zrealizowanej przez CBOS w 2000 roku. Wykorzystaj w tym celu [sortowanie przez scalanie](#) (*merge sort*).

Procentowe wyniki ankiety „Czy pana/pani zdaniem równość w społeczeństwie powinna czy też nie powinna oznaczać, że wszyscy obywatele mają równe szanse zdobycia wykształcenia i osiągnięcia wysokiej pozycji społecznej, o ile mają odpowiednie zdolności i chęci?”, zrealizowanej w lutym 2000 roku przez CBOS:

- 67% odpowiedziało: „Zdecydowanie tak”.
- 21% odpowiedziało: „Raczej tak”.
- 2% odpowiedziało: „Trudno powiedzieć”.
- 3% odpowiedziało: „Zdecydowanie nie”.
- 7% odpowiedziało: „Raczej nie”.

Specyfikacja:

Dane:

- `dane[ ]` – tablica liczb naturalnych

Wynik:

- Na standardowym wyjściu wyświetlane są wszystkie elementy tablicy `dane`, posortowanej w porządku niemalejącym, oddzielone pojedynczym znakiem spacji.

Twoje zadania

1. Wypisz elementy tablicy posortowane niemalejąco za pomocą metody sortowania przez scalanie.

```
1 #include <iostream>
2
3 using namespace std;
4
5 int dane[] = {67, 21, 7, 3, 2};
6 int liczbaElementow = 5;
7
8 void sortowaniePrzezScalanie(int indeksPoczatkowy, int
   indeksKoncowy) {
9     // tutaj zaimplementuj sortowanie przez scalanie
10 }
11
12 int main()
13 {
```

```
1
```

Polecenie 1

Porównaj swoje rozwiązanie z przedstawionym w filmie.



Sortowanie tablicy metodą przez scalanie

Algorytm i jego realizacja w języku C++



Film dostępny pod adresem </preview/resource/RsjejBvogjsQu>

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Film nawiązujący do treści materiału: Sortowanie tablicy metodą przez scalanie.

Kod programu zaprezentowanego w filmie:

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Plik o rozmiarze 1.26 KB w języku polskim

Słownik

rekurencja

metoda programowania w dowolnym języku, polegająca na wywołaniu przez funkcję samej siebie; rekurencję można porównać do pętli, ponieważ jest ona wywoływana określoną liczbę razy (do momentu, gdy zajdzie warunek stopu)

sortowanie przez scalanie

podzielenie sortowanej struktury danych na dwie części, następnie na rekurencyjnym sortowaniu każdej z tych części i ponownym scaleniu już posortowanych części w jedną całość

Prezentacja multimedialna

Sortowanie przez scalanie możemy zaimplementować na wiele sposobów. W prezentacji pokażemy kod, w którym wydzielono oddzielną funkcję służącą do scalania tablicy, po zakończonym procesie podziału. Aktualnie przetwarzane dane będziemy przechowywać w tymczasowych tablicach, usuwanych z pamięci po zakończeniu działania funkcji rekurencyjnej. Sortowaną tablicę będziemy przekazywać jako jeden z parametrów wywołania funkcji.

Polecenie 1

Zapoznaj się z prezentacją. Przygotuj specyfikację dla przedstawionego problemu.

Materiał audio dostępny pod adresem:

<https://zpe.gov.pl/b/PPpC0G82I>

1

Implementowanie algorytmu w C++ zaczniemy od stworzenia funkcji rekurencyjnej `sortuj`, która jako argumenty przyjmuje tablicę do posortowania, indeks początku oraz indeks końca analizowanej właśnie części.

Funkcja przestanie tworzyć kolejne wywołania rekurencyjne w momencie, w którym przyjmie ona jako argument tablicę jednoelementową. W takiej sytuacji funkcja kończy działanie.

2

Materiał audio dostępny pod adresem:

<https://zpe.gov.pl/b/PPpC0G82I>

Jeżeli w badanej części będzie tylko jeden element, `indeks p` będzie równy `indeks k`.

Tak długo, jak w tablicy znajdować się będzie więcej niż jeden element, będziemy wykonywać opisane na poprzedniej stronie kroki dla wywołań funkcji rekurencyjnej.

Materiał audio dostępny pod adresem:

<https://zpe.gov.pl/b/PPpC0G82I>

3

Wspomniane kroki to: podział na dwie części, podział lewej części, podział prawej części oraz złączenie części.

Łączenie części wymaga zapisania relatywnie dużej ilości kodu, więc do operacji tej napiszemy osobną funkcję `scal`.

4

Materiał audio dostępny pod adresem:

<https://zpe.gov.pl/b/PPpC0G82I>

Zadeklarujemy więc funkcję `scal` w następujący sposób:

Oprócz tablicy, zmienne, które przyjmie funkcja to:

- `indeksp` – odpowiada pierwszemu indeksowi lewej części,
- `indeksgraniczny` – ostatni indeks lewej części,
- `indeksk` – ostatni indeks prawej części.

Wśród zmiennych brakuje pierwszego indeksu prawej części – to dlatego, że jest to indeks

o jeden większy od tego w zmiennej indeksgraniczny – jeśli będziemy go potrzebować, posłużymy się wartością tej właśnie zmiennej.

5

Materiał audio dostępny pod adresem:

<https://zpe.gov.pl/b/PPpC0G82I>

Aby swobodnie zamieniać elementy miejscami, stworzymy kopie lewej oraz prawej części tablicy. Dzięki kopiowaniu nie będziemy nadpisywać danych w tablicy. Aby stworzyć kopie, najpierw musimy określić rozmiar lewej oraz prawej części.

|

6

Materiał audio dostępny pod adresem:

<https://zpe.gov.pl/b/PPpC0G82I>

Znając już rozmiar lewej oraz prawej części, możemy zainicjować tablice, które będą przechowywać kopie elementów znajdujących się w tych właśnie częściach tablicy.

|

Kolejnym krokiem będzie wypełnienie tablic wartościami z lewej oraz prawej części.

7

Materiał audio dostępny pod adresem:

<https://zpe.gov.pl/b/PPpC0G82I>

Wypełniamy nowe tablice wartościami z odpowiednich indeksów z tablicy.

8

Materiał audio dostępny pod adresem:

<https://zpe.gov.pl/b/PPpC0G82I>

Zanim przejdziemy do scalania, musimy zainicjować kilka zmiennych pomocniczych.

Pierwsze dwie zmienne będą odpowiedzialne za przechowywanie bieżącego indeksu każdej z części. Jeżeli przeniesiemy element z lewej części, zmienna `teraz_indekslewej` zostanie powiększona o jeden i będzie wskazywać na pierwszy nieprzeniesiony jeszcze element lewej części.

Zmienna `teraz_indeksorg` wskazuje miejsce w oryginalnej tablicy, gdzie zostanie umieszczony kolejny element.

Materiał audio dostępny pod adresem:

<https://zpe.gov.pl/b/PPpC0G82I>

9

Mając już kopie tablic oraz zmienne pomocnicze, możemy przejść do scalania części. Zaczniemy od porównania elementów dwóch części. Dopóki obie części mają przynajmniej po jednym elemencie, porównujemy je i przenosimy większy.

Sprawdzimy teraz, czy pierwszy element lewej części jest większy lub równy prawemu. Jeżeli jest to pierwszy element z lewej części, zostaje

przeniesiony na miejsce `teraz_indeksorg`,
w przeciwnym wypadku trafia tam pierwszy
element prawej części.

10

Materiał audio dostępny pod adresem:

<https://zpe.gov.pl/b/PP Pc0G82I>

Scalanie jest już gotowe:

Gdy w jednej z części skończą się elementy,
wszystkie pozostałe elementy z drugiej zostają
przeniesione w takiej kolejności, w jakiej się
znajdują.

Materiał audio dostępny pod adresem:

<https://zpe.gov.pl/b/PP Pc0G82I>

11

Wywołajmy więc naszą funkcję i sprawdźmy jej
działanie.

12

Materiał audio dostępny pod adresem:

<https://zpe.gov.pl/b/PP Pc0G82I>

Kompletny program wygląda następująco:

Polecenie 2

Opracuj notatkę zawierającą najważniejsze informacje przedstawione w prezentacji.

Sprawdź się

Pokaż ćwiczenia:   

Ćwiczenie 1



Napisz program, który na podstawie tablicy `zbior` stworzy nową tablicę i nazwie ją `kopiaZbioru`. Program powinien skopiować tablicę, zaczynając od elementu oznaczonego indeksem `a` i kończąc na elemencie o innym indeksie `b`. Przetestuj działanie programu dla indeksu początkowego 3 oraz indeksu końcowego, który jest o 3 mniejszy niż długość tablicy.

Działanie programu przetestuj dla tablicy:

```
1 int zbior[16] = { 43, 6, 76, 87, 54, 654, 432, 657, 765, 543, 6
```

Nowa tablica powinna składać się ze wszystkich elementów, poza trzema pierwszymi oraz trzema ostatnimi.

Specyfikacja:

Dane:

- `n` – liczba naturalna
- `a` – liczba naturalna
- `b` – liczba naturalna
- `zbior[ ]` – `n`-elementowa tablica liczb naturalnych

Wynik:

Na standardowym wyjściu wyświetlane są wszystkie elementy tablicy `kopiaZbioru`, utworzonej zgodnie z treścią zadania, oddzielone pojedynczym znakiem spacji.

Twoje zadania

1. Program tworzy kopię zbioru bez trzech pierwszych oraz trzech ostatnich elementów oraz wypisuje zawartość zbioru.

```
1 #include <iostream>
2 using namespace std;
3
4 int main ()
5 {
6     int zbior[16] = {43, 6, 76, 87, 54, 654, 432, 657, 765,
7     543, 654, 7657, 987, 4, 43, 65};
8
9     // Tutaj dodaj kod.
10
11     // Do wypisywania, użyj poniższej linii
12     // for (int i = 0; i < 10; i++) cout << kopiaZbioru[i] <<
13     " ";
14 }
```

1

Ćwiczenie 2



Napisz program, w którym scalisz dwie posortowane tablice w jedną (również posortowaną).
Wypisz utworzoną tablicę, oddzielając jej elementy znakiem spacji.

Swój program przetestuj dla następujących tablic:

```
1 int tablica1[6] = { 4, 65, 432, 543, 654, 7654 }
2 int tablica2[7] = { 2, 23, 45, 65, 75, 432, 9211 }
```

Specyfikacja:

Dane:

- n – liczba naturalna
- m – liczba naturalna
- `tablica1` – n -elementowa posortowana niemalejąco tablica liczb naturalnych
- `tablica2` – m -elementowa posortowana niemalejąco tablica liczb naturalnych

Wynik:

Na standardowym wyjściu wyświetlane są wszystkie elementy posortowanej niemalejąco tablicy, będącej wynikiem scalenia tablic wejściowych. Elementy są oddzielone pojedynczym znakiem spacji.

Twoje zadania

1. Program łączy dwie tablice w jedną.

```
1 #include <iostream>
2 using namespace std;
3
4 int main ()
```

```
5 {  
6     int tablica1[6] = { 4, 65, 432, 543, 654, 7654 };  
7     int tablica2[7] = { 2, 23, 45, 65, 75, 432, 9211 };  
8  
9     // Tutaj dodaj kod  
10  
11     // Do wypisywania użyj funkcji cout  
12 }
```

```
1
```

Ćwiczenie 3



Napisz program, w którym posortujesz malejąco daną tablicę `tablica`, używając algorytmu sortowania przez scalanie. Wypisz jej elementy, oddzielając je znakiem spacji.

Działanie programu przetestuj dla tablicy:

```
1 int tablica[31] = { 4, 656, 65, 43, 65, 876, 543, 756, 435, 654
```

Specyfikacja:

Dane:

- `n` – liczba naturalna
- `zbior[ ]` – `n`-elementowa tablica liczb naturalnych

Wynik:

Na standardowym wyjściu wyświetlane są wszystkie elementy posortowanej malejąco tablicy `tablica`, oddzielone pojedynczym znakiem spacji.

Twoje zadania

1. Program sortuje tablicę, używając algorytmu sortowania przez scalanie.

```
1 #include <iostream>
2 using namespace std;
3
4 int main ()
5 {
6     int tablica[31] = { 4, 656, 65, 43, 65, 876, 543, 756,
7                        435, 654, 534, 234, 756, 432, 765, 543, 987, 423, 123, 423,
8                        654, 756, 534, 6755, 534, 4756, 65, 4, 654, 54, 75 };
9
10    // Tutaj dodaj kod.
```

```
10 // Do wypisywania, użyj poniższej linii
11 // for (int i = 0; i < 31; i++) cout << tablica[i];
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

Dla nauczyciela

Autor: Maurycy Gast

Przedmiot: Informatyka

Temat: Sortowanie przez scalanie w języku C++

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

3) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

I + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

1) zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach oraz algorytmy:

e) sortowania ciągu liczb przez scalanie,

3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

b) rekurencję (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali),

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Przeanalizujesz, czym jest rekurencja oraz metoda „dziel i zwyciężaj”.
- Scharakteryzujesz działanie algorytmu sortowania przez scalanie.
- Napiszesz program wykorzystujący ten algorytm.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimediu i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Sortowanie przez scalanie w języku C++”. Uczniowie mają zapoznać się z treściami w sekcji „Film samouczek” i wykonać obliczenia na podstawie dołączonych danych.

Faza wstępna:

1. Przedstawienie tematu i celów zajęć.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. Uczniowie analizują film z sekcji „Film samouczek” wyświetlony na tablicy. Wybrany uczeń czyta treść polecenia nr 1: „Napisz program sortujący wyniki ankiety dotyczącej równości w życiu społecznym, zrealizowanej przez CBOS w 2000 roku. Wykorzystaj w tym celu sortowanie przez scalanie (merge sort)” i omawia przykładowe rozwiązanie postawionego problemu.
2. **Praca z multimediami.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna”. Uczniowie wspólnie zapoznają się z jego treścią. Zapisują ewentualne problemy i pytania. Po czym następuje dyskusja, w trakcie której nauczyciel wyjaśnia niezrozumiałe treści.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 1: „Napisz program, w którym utworzysz kopię tablicy `zbior` bez pierwszych oraz ostatnich trzech elementów. Nazwij ją `kopiaZbioru`. Następnie wypisz jej elementy, oddzielając je znakiem spacji”, a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.
4. Liga zadaniowa – uczniowie pracując w parach, wykonują ćwiczenie nr 2: „Napisz program, w którym scalisz dwie posortowane tablice w jedną (również posortowaną). Wypisz utworzoną tablicę oddzielając jej elementy znakiem spacji” z sekcji „Sprawdź się”, a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.

Faza podsumowująca:

1. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy w zakresie programowania w języku C++.

Praca domowa:

1. Uczniowie wykonują ćwiczenie nr 3: „Napisz program, w którym posortujesz malejąco daną tablicę `tablica`, używając algorytmu sortowania przez scalanie. Wypisz jej elementy, oddzielając je znakiem spacji” z sekcji „Sprawdź się”.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Wskazówki metodyczne:

- Treści w sekcji „Film samouczek” można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.